VIPER

April-May-June 1983
Volume 5, Number 1

Journal of the VIP Hobby Computer Assn.

The VIPER was founded by ARESCO, Inc. in June 1978

****

PIPS for VIPS IV

This issue of VIPER begins the serialization of Tom Swan's PIPS for VIPS IV. Tom has had the manuscript completed for some time and has given VIPHCA permission to publish it in VIPER. Tom's manuscript arrived about a week before I was going to take an issue to the printer. I was completely floored by the amount of the material. And, as you might expect, it was as well written as the other of Tom's PIPS for VIPS. So, it was "Stop the Presses!" and let's get this into print. That's why this issue has been delayed. There's no point in pretending that this is an April-May issue, so I added June, and that way the production schedule will be bumped up by one month. Once things get rolling with PIPS, VIPER will be out every two months.

You'll also notice the different print format. Most VIPHCA people put their VIPER issues into a 3-ring binder, so I figured we might just as well print in that format. It will also make the printing of VIPER a bit more flexible, since only a single sheet of paper (2 sides) needs to be formatted at one time. And the printing serivce is much quicker, since most people use 8 1/2 by 11 printing and the printer can run off the copies without having to reset the offset press for 11 by 17. And I can even add late items via a photocopy print run. Let me know how this works out for you. I hope it'll be OK, but let me know if there are problems. There will also be cassettes of this PIPS series available. Details later.

73
Ray

5.01.00

News and Views

Well, getting this issue to bed was a bit of a job, and I thank all of you out there for your patience. While Tom Swan's PIPS series is directed to the VIP, there is still much of value for ELF owners. Some of the machine language routines can be used on machines other than the VIP, but even those that cannot are so well documented as to suggest ways to write your own routines. I think that the greatest value of Tom's material is in the way it is so well commented. It's really tough trying to figure out what a programmer means when you see "6100 ; V1=00." Heck, 6100 tells you that much! What's really important is to know that V1 is a display co-ordinate, or whatever.

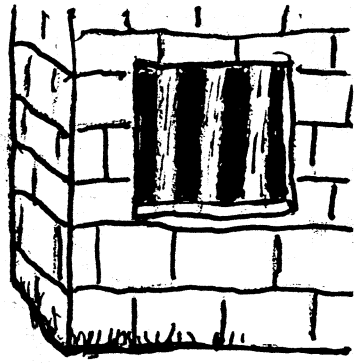
Don King of 4269 Trenton, Detroit MI 08610, has called to say that he, like quite a few of or VIPHCA members, has developed a memory expansion board for the VIP. Don will have a bare board and a fully populated version available. You should check with him about latest prices. You might also check out some of the other memory boards made by other VIPHCA members, and advertised in this issue. Large amounts of available memory open up some rather interesting programming possibilities for the VIP. For one, you could do some text processing. Or you could have an assembler/disassembler in memory at all times, ready to be called upon. Or you could have a dozen games all loaded in and called up from a main menu. The boards which permit mixing read/write RAM and EPROM are especially attractive, since you can store your favorite programs or routines in EPROM, ready for instant use. And I wouldn't put it past some one of you to whip up a 'CALC type program for the VIP to make use of a large (32K or so) block of memory!

The first program in PIPS IV is COS-MELODEON, which is printed in its entirety. You will note that it requires a character set and messager routines, from the earlier PIPS series, as well as a Simple Sound board. There will be cassettes available of the whole PIPS IV series of programs, but if you want the COS-MELODEON one right away, send in $2 and I'll mail it out will all the needed routines. Otherwise, you'll have to wait a bit for the complete set, although the full set will be less expensive on a per-program basis.

In any event, PIPS IV is now available to us, and I hope you enjoy the material. Thank you very much, Tom. Also, I still have a CHIP-8 assembler, written in BASIC, by Bill Lindley and some of Paul Piescik's CSOS programs, which are pending for future VIPER issues. I hope you'll find them all worthwhile.

73, Ray

ESCAPE FROM PRISON



In our all new high resolution, scrolling text adventure game, you find yourself in a locked prison cell and you must escape! To do so, you type in two-word commands consisting of verb+object, and you will find yourself wandering from room to room, manipulating objects to help you reach your goal. A must for people who like fantasy role-playing games like "Dungeons and Dragons" by TSR corp. Requires 4K VIP with ASCII keyboard.

-Escape From Prison- $8.95, shipping included.

.....

Venture back to the days of King Arthur in Quest of the Enchanted Sword, our original VIP adventure! The game starts in a forest in Camelot, shortly after King Arthur's death, and you must go from there...

On the flip side of the cassette are three "mini" adventures, written for a 4K VIP with Tiny Basic. They are similar to adventures, and the object is to get a high score based on the number of monsters you kill and the amount of treasure you get.

The cassette requires a 4K VIP with an ASCII keyboard, and Tiny Basic is optional.

-Quest of the Enchanted Sword-$8.95, shipping included

.....

Run through a maze and eat up all the dots before the monsters chasing you catch up with VIP-MAN, our arcade-type game! Amazing high resolution graphics, color, sound effects, and four-digit scoring make this a game worth getting. Requires 4K VIP, color and sound boards optional.

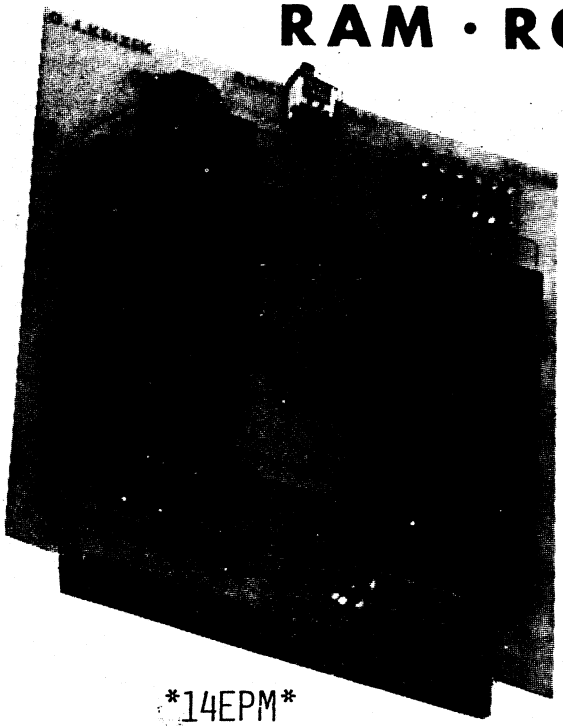
-VIP-MAN-$9.95, shipping included.

.....

Send orders to:

VIP Adventure, Unltd.
168 Pond St.
Sharon, MA. 02067

RAM · ROM for VIP



14EPM

THE 14EPM GIVES YOU ALMOST INSTANT VP-701 FLOATING POINT BASIC. JUST SWITCH TO RUN, PRESS "1" ON HEX KEYPAD, AND PRESS "W" OR "C" ON YOUR ASCII KEYBOARD TO SELECT WARM OR COLD START. USES 5V 2716s.

BARE BOARD WITH DATA...$ 39.00
A&T LESS EPROMS.....$ 59.00
A&T W/ EPROMS PROGRAMED WITH DATA PROVIDED BY YOU*...$119.00
A&T W/ RCA'S VP-701**...$139.00

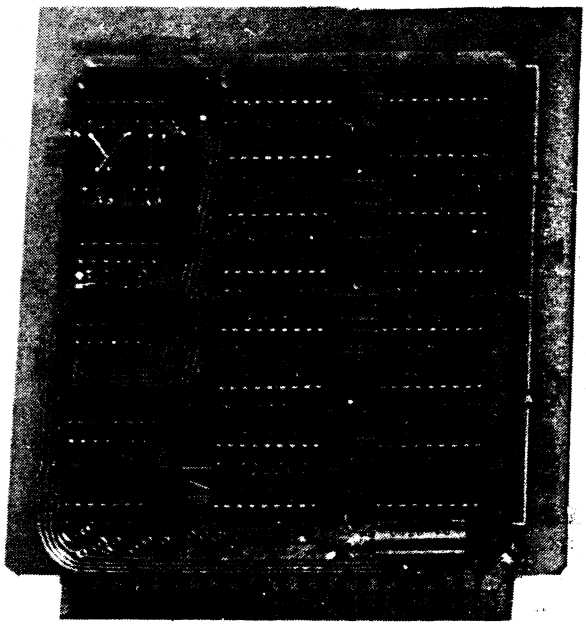
*DATA MUST BE ON VIP COMPATIBLE CASSETTE IN 2K BLOCKS.

**STATE YOUR HIGHEST RAM ADDRESS AND IF YOU WANT COLOR COMMANDS.

THE 8KRAM CARD GIVES YOU TWO 4K BLOCKS OF STATIC RAM ADDRESSABLE TO ANY 4K BLOCK IN VIP MEMORY. USES POPULAR 2114 RAMS.

BARE BOARD WITH DATA.$ 49.00
A&T LESS RAMS.....$ 79.00
A&T W/ RAMS.....$129.00

NOTE: 8KRAM CARD REQUIRES APROX 600MA FROM 5V LINE.



8KRAM

****
* PACKAGE SPECIAL *
****

2-8KRAM, 1-14EPM BARE BOARDS WITH DATA.....$ 99.00
2-8KRAM, 1-14EPM A&T W/ RAMS AND VP-701 BASIC.....$349.00

(CALIF. RESIDENTS ADD SALES TAX.)

G. J. KRIZEK
722 N. MORADA AVE.
WEST COVINA, CA.
91790

COS-MELODEON
by Tom Swan

In the woods above the Seglock Creek in Pennsylvania is a cabin that my grandfather built by hand when my father was in college. There, in a corner of a wood smokey room, they keep a beautiful working player piano with foot pedal pumps and a speed lever you control with your knee.

Playing that old, slightly out of tune one-man orchestra one day gave me the idea for this program, COS-MELODEON. There are no instructions for running the program except that you must have an RCA Simple Sound Board plugged into the righthand I/O slot of your VIP computer and at least 4K of memory. (Remember to turn off your computer BEFORE installing or removing the sound board.)

Load "C" pages from tape starting at 0000, flip to run and sit back. You don't even have to pedal -- COS-MELODEON is one of them "new-fangled" automatic pie-annies!

PROGRAM DESCRIPTION

The first thing you may want to experiment with is COS-MELODEON's song. Like a player piano, you may change "rolls" to play new tunes by entering pitch and duration information into memory. In addition, you must also tell COS-MELODEON which key to "depress" while playing a note. The paper roll is only for effect and will scroll up to your new song without any help.

Though CHIP-8 programs may only normally address or reference memory located in the initial 4K (4096 bytes) of computing space, COS-MELODEON permits lengthy songs to be entered starting at hexadecimal $1000 and continuing as far into memory as you like. Of course, you must have the extra memory installed in your computer in order to use this feature.

Each note of a song is kept in a table in memory with each note requiring three bytes in the following format:

BYTE #1 = PITCH .
BYTE #2 = DURATION
BYTE #3 = KEY CODE

Changing the entries in the table will change the notes played by COS-MELODEON. The first byte of a table entry represents the pitch of the note and may be any value from hex $01 of $FE for a full range of tones. Entering a pitch of 00 will cause a rest note or "silent" note to be played. Entering a pitch of FF ends the song.

On the following page you will find a note chart which gives the hexadecimal values corresponding to COS-MELODEON's keyboard range. The Simple Sound Board manual contains many more hex codes for additional notes that may be programmed. After the hex value for a note in the chart, a second value is given. This is the key code which goes into byte #3 of an entry in the song table. The key code tells COS-MELODEON which key to "depress" and must be in the range of 00 to #34.

NOTE CHART - COS-MELODEON

(OCTAVE)	NOTE	HEX	KEY CODE
3	C	D2	00
	C#	C6	1F
	D	BB	01
	D#	B0	20
	E	A6	02
	F	9D	03
	F#	94	21
	G	8B	04
	G#	84	22
	A	7C	05
	A#	75	23
	B	6F	06
4	C	68	07
	C#	62	24
	D	5D	08
	D#	58	25
	E	53	09
	F	4E	0A
	F#	49	26
	G	45	0B
	G#	41	27
	A	3E	0C
	A#	3A	28
	B	37	0D

(OCTAVE)	NOTE	HEX	KEY CODE
5	C	34	0E
	C#	31	29
	D	2E	0F
	D#	2B	2A
	E	29	10
	F	26	11
	F#	24	2B
	G	22	12
	G#	20	2C
	A	1E	13
	A#	1D	2D
	B	1B	14
6	C	19	15
	C#	18	2E
	D	16	16
	D#	15	2F
	E	14	17
	F	13	18
	F#	12	30
	G	11	19
	G#	10	31
	A	0F	1A
	A#	0E	32
	B	0D	1B
7	C	0C	1C
	C#	0B	33
	D	--	1D
	D#	0A	34
	E	--	1E

Even if you program a note not corresponding to one of COS-MELODEON's 53 keys, you must still specify one of the keys to be depressed by using a valid key code in byte #3.

Byte #2 of a note entry determines the length of time that the note or rest will be played. A value of $6 correspponds roughly to a 1/16th note with $C = 1/8th, $18 = 1/4 notes, etc., incrementing by values of 6 for longer and longer notes. A value of 0 is the shortest possible note with $FF the longest. You may fiddle, so to speak, with the relation of timing for wide speed variations.

To program a melody, sheet music may be used. To produce a stacatto effect you will have to combine rests and notes or the tones will all run together. (Stacatto is the term for choppy, clear cut notes. Legato means the opposite and refers to notes that blend into each other.) Due to the screen animation, COS-MELODEON notes will normally sound slightly "stacatto." Because of the display instructions which are intertwined with note playing, timing will be different from the suggestions given in the Simple Sound manual.

Rest notes (pitch byte = 00) also require duration information and tend to "play" a little quicker than actual notes. This is because the instructions for "depressing" and "releasing" keys are skipped for rests and the loop executes a bit faster. Try adding 1/4 to 1/3 of a note value for corresponding rests. Key codes for rest notes are ignored and may be set to any value, but a value must still be included. Rest notes are three bytes long, even though the third byte is ignored.

The $FF byte marks the end of a song and prevents the unworldly sounds of COS-MELODEON attempting to orchestrate your operating system or other memory patterns following the end of the song table.

Normally, a song table begins at $0A03 (NOT 0A00) and may extend right up to the beginning of the four page display refresh area at $0C00. If your song has more than 167 notes and rests, however, you will need to modify COS-MELODEON as follows:

To begin a song list at $1000, which is just above CHIP-8's addressing range, change location $0794 to $AFFD. Song lists always begin three bytes beyond the setting of the "I" pointer set by this instruction. ($AFFD +3 = $1000) Two special machine language subroutines (MSL's) named "INIT" and "NEXT" permit "I" to go beyond its normal limit of $0FFF. (Note that it would also be possible to begin a song list at $0A00 by programming the instruction $A9FD at location $0794.) Your song may now fill all available RAM starting from $1000. There is no easy way to begina song list at a location higher than $1000, however.

The program is so fully documented that there is no need to discuss each individual part, as I have for some programs. Though lengthy, most of the code is responsible for drawing the piano, displaying the title, etc., and should not be too hard to follow.

All routines except the main loop were written as subroutines. This was done even when a cection is used only once, as in the case of "DOORS" which causes the piano doors to be drawn.

Normally, subroutines are used when a section will be called many times from different places in a program. But they are also helpful when designing a program by allowing the programmer to write small pieces or modules instead of tackling the whole idea at once. Each module should stand alone, that is, it should do a particulat job without a lot of complications, then end when that job is done. When all the modules have been written, tested, rewritten, and retested, the final program will usually go together as easy as "uno, dos, diez." (That's "one, two, three" in binary and Spanish. "Spinary?" "Banish?")

To see what I mean, enter the subroutine call for "DOORS" at the beginning of the main loop followed by a "jump to self" command to halt the program. Do this AFTER loading COS-MELODEON from tape. This is what you enter:

```
0500  25B2  DOORS  ;Do sub to draw doors
0502  1502         ;Stop by "jump to self"
```

Program the above using the VIP operating system "write memory mode," then flip to run. You should see COS=MELODEON's doors being drawn on the screen. Likewise, you an draw just the piano body. Try entering $2802 then $1502 at $0500 and $0502 and running. In fact, just about any of COS-MELODEON's main subroutines (those that are listed with an asterisk boxed heading) may be individually run in this manner.

When trying out new songs, simply call the PLAY subroutine by entering a 2794 instruction at $0500. (Again, remeber the 1502 jump at 0502 or the rest of the program will run after the song plays!) Though the comical video embellishments of a "ghost COS-MELODEON" may distract you, this is a handy way to test a song list without waiting for the title and preliminary animations to finish each time. (For a laugh, try this even if you don't have a new song to test out!)

COS-MELODEON was, in fact. written in this very way with the front part of the main loop, the part that calls each subroutine in the correct order, being constructed last. Rather than try to write programs in one long sequential line (and "Oh no!" you forgot a subtract instruction three pages back!), it will help you to organize the task into small parts. Get each part going properly, and the rest will fall into place.

* * *

Two of COS-MELODEON's machine language subroutines (MLS's) were written so that they may be taken out and used in your own programs. (Provided you do not intend to sell them as your own original work, of course.)

The first is the TOGGL (short for Toggle) MLS at 0744 to 074E. This routine's effect is further described in the early 1980 issues of the VIPER newsletter. What it does is to change a byte inside the CHIP-8 interpreter at location 00AC from 00 to EC or from EC to 00 each time the MLS is called. When the byte is 00AC is set to EC, CHIP-8's display instruction DXYN runs at super fast speed. During the firing of MICROMAN's laser, CHIP-8 is too slow for the effect I wanted, and TOGGL takes care of that.

If you use TOGGL in your own programs, it may be loaded into any place in memory. In other words, TOGGL is "relocatable" and contains no branch instruction. For TOGGL to work, your program must use a RAM version of CHIP-8 loaded into the beginning of memory. The CHIP-8 listing in RCA's VIP manual, as well as the listing here, other PIPS volumes, and issues of the VIPER will work with TOGGL. ROM versions of CHIP-8 will NOT allow TOGGL to operate, since TOGGL actually modifies the interpreter when called.

One caution: when DXYN'ing at high speed, complex patterns may occasionally break up for a moment and appear ragged. TOGGL is best used to speed up single dot or very simple animations such as MICROMAN's laser or the ball in a game of Wipe Off. Page switching (see next issue) may also be used to smooth out the faster display.

The second MLS you may find useful is SCROL (for scroll, what else?) located at $0930 to $095D. The SCROL sub is what makes COS-MELODEON's paper roll appear to roll up as the tune plays. As long as the area you want to scroll is no more than two bytes wide, any display area may be scrolled up by this routine. The instruction at 0930 may be increased to extend the vertical range of SCROL. The starting addresses set into RE and RF by the instructions at $0933 to $093C may also be adjusted to reflect the tup addresses of the SCROL "window."

Every time SCROL is called, everything in the display "window" is moved up one bit. The routine itself is fast enough to scroll as much as four pages of information so that all of the bits appear to move at once, but to scroll that much data would require modifying the subroutine more than just a little.

* * *

Just a final note (now THAT wasn't even funny!) -- if you send in your songs to the VIPER, we can all start building a lifrary of "rolls" for COS-MELODEON. First version of the Brandenberg Concerti wins COSMAC Composer of the Year Award.

COS-MELODEON

VARIABLE ASSIGNMENT

V0 - Various uses. Display coordinates. Loops. Indexes.
V1 - " " " " " "
V2 - " " " " " "
V3 - " " " " " "
V4 - Not used
V5 - " "
V6 - " "
V7 - " "
V8 - " "
V9 - " "
VA - " "
VB - " "
VC - VX for some display instructions (DXYN's)
VD - VY " " " " "
VE - Outputs tone in WOOPS sub
VF - Utility. Passes timer value to TIMER subroutine

MEMORY MAP (4K)

0000 - 029F -- CHIP-8 hi-res interpreter with MESSAGER and I/O *
02A0 - 02EF -- CHIP-8 work space and stack
02F0 - 02FF -- CHIP-8 variables
0300 - 04FF -- ASCII character set bit patterns
0500 - 07E7 -- COS-MELODEON program (CHIP-8)
07E8 - 07FF -- Available space
0800 - 08A7 -- COS-MELODEON program (CHIP-8)
08A8 - 08FF -- Available space
0900 - 09BD -- COS-MELODEON program (MLS's and tables)
09BE - 09FF -- Available space
0A00 - 0BFF -- COS-MELODEON song table
0C00 - 0FFF -- 4-page display refresh

*Note -- CHIP-8 variables and stack on page $02

PROGRAM LISTING

```
        ; ****************
        ; * COS-MELODEON *
        ; ****************

        ; C. 1979 T. Swan

        ; Requires:

        ;       Hi-res CHIP-8
        ;       with MESSAGER
        ;       and I/O

0500  BEGIN: 2654   TITLE -- Do sub to display title
  02         6FFF  ;VF=FF -- Set VF to timer value
  04         2564   TIMER -- Do sub to delay for time in VF
  06         269A   FIXIT -- Do sub, fix letter via MICROMAN
  08         256E   KEYBD -- Do sub to draw keyboard
  0A         25B2   DOORS -- Do sub to draw doors
  0C         2802   PIANO -- Do sub to draw body of piano
  0E         2602   SYMBL -- Do sub to display price & GCLEF

0510         6F40  ;VF=40 -- Set VF to timer value
  12         2564   TIMER -- Do sub to delay for time in VF
  14         2768   OPEN  -- Do sub to open doors
  16         6F40  ;VF=40 -- Set VF to timer value
  18         2564   TIMER -- Do sub to delay for time in VF
  1A         6018  ;V0=18 -- V0, V1 are VX VY to display
  1C         615C  ;V1=5C --   the initial "paper" line
  1E         A562   TRS5  -- Set "I" to address of bit pattern

0520         D011  ;SHOW  -- Show top of paper at V0, V1
  22         7008  ;V0+8  -- Add 8 to V0 (VX)
  24         D011  ;SHOW  -- Show rest of paper (2 bytes wide)
  26         6020  ;VF=20 -- Set up loop count of 20 in V0
  28  TRS1:  0930  ;SCROL -- Call MLS.  Scroll paper up one bit
  2A         6F08  ;VF=8  -- Set VF to timer value
  2C         2564   TIMER -- Do sub to delay for time. in VF
  2E         70FF  ;V0-1  -- Subtract 1 from loop count

0530         3000  ;SK=0  -- Skip when loop count in V0 equals zero
  32         1528   TRS1  -- Jump to continue initial scrolling
  34         2794   PLAY  -- Do sub, play song
  36         6040  ;V0=40 -- Set up loop count in V0
  38         6110  ;V1=10 -- Set up second loop in V1
  3A  TRS3:  0930  ;SCROL -- Call MLS to scroll to end of paper
  3C         6F08  ;VF=8  -- Set VF to timer value
  3E         2564   TIMER -- Do sub to delay for time in VF

0540         70FF  ;V0-1  -- Subtract 1 from loop in V0
  42         71FF  ;V1-1  -- Subtract 1 from loop in V1
  44         3100  ;SK=0  -- When V1=0, skip next jump
```

```
0546          153A    TRS3   -- Jump to scroll paper
  48          4000   ;SK≠0   -- Skip as long as V0 is not zero
  4A          155C    TRS4   -- Jump to exit this loop
  4C          6C18   ;VC=18  -- Set VC, VD to X, Y coordinates of
  4E          6D5C   ;VD=5C  --   bottom of piano paper

0550          A562    TRS5   -- Set "I" to bit pattern
  52          DCD1   ;ERASE  -- Show to erase two bytes at
  54          7C08   ;VC+8   --   bottom edge of paper.  Thus
  56          DCD1   ;ERASE  --   further scrolling will erase page.
  58          8100   ;V1=V0  -- Reset loop count V1 to equal V0
  5A          153A    TRS3   -- Jump to end scrolling
  5C  TRS4:   277E    SHUT   -- Do sub to shut doors
  5E          2874    ENCOR  -- Do sub.  MICROMAN's encore
0560  HALT:  1560    HALT   -- End of program.  (Loop forever)
                             -- Change to 1510 to replay continuously
  62  TRS5:  FF00           -- Bit pattern for "paper roll"

      ; **********
      ; * TIMER *
      ; **********

      ; INPUT:
      ;       VF = TIMING

0564  TIMER: FF15   ;T=VF   -- Set CHIP-8 timer to VF value
  66  TIME1: FF07   ;VF=T   -- Set VF to current timer value
  68          3F00   ;VF=0? -- If VF=0, then skip to exit
  6A          1566    TIME1  -- Else jump to loop and retest timer
  6C          00EE   ;RET    -- Return from subroutine

      ;END TIMER

      ; ************
      ; * KEYBOARD *
      ; ************

      ; VC, VD = VX, VY
      ; V0 = I INDEX

      ; INPUT: NONE
      ; OUTPUT:
      ;       KEYS DRAWN
      ; CALLS:
      ;       TIMER

      ; WHITE KEYS
      ; VX = 00 to 3E
      ; VY = 70 to 7F

  6E  KEYBD: 6C00   ;VC=0   -- Set initial VX for drawing keyboard
0570          A5A0    KEYB5  -- Set "I" to bits for white keys
  72  KEYB1: 6D70   ;VD=70  -- Set VY coordinate in VD
```

```
0574  KEYB2: DCD1   ;SHOW  -- Show single bit
  76         7D01   ;VD+1  -- Add 1 to Y coordinate
  78         3D80   ;SK=80 -- If VD (VY)=80, then skip next instruction
  7A         1574    KEYB2 -- Jump, draw one vertical white key
  7C         7C01   ;VC+1  -- Add 1 to X coordinate
  7E         3C3F   ;SK=3F -- If VC (VX)=3F, then skip next instruction
0580         1572    KEYB1 -- Jump to draw all white keys

       ; BLACK KEYS
       ; VX = 00 to 3E
       ; VY = 73 to 7A

  82         6C00   ;VC=0  -- Reset VX coordinate in VC
  84   KEYB3: 6000   ;V0=0  -- Set V0 as index to black key patterns
  86         A5A2    KEYB6 -- Set "I" to black key bit patterns
  88   KEYB4: 6D73   ;VD=73 -- Set VY coordinate for black keys
  8A         F01E   ;I+V0  -- Index "I" to proper pattern by adding V0
  8C         6F10   ;VF=10 -- Set VF to timer value
  8E         2564    TIMER -- Do sub to delay by time in VF

0590         DCD8   ;SHOW  -- Show one set of black keys
  92         7C07   ;VC+7  -- Add 7 to VX coordinate
  94         7008   ;V0+8  -- Add 8 to V0 index
  96         4C3F   ;SK≠3F -- If VC ≠ end of keyboard, skip next
  98         00EE   ;RET   -- Return.  All keys drawn
  9A         3010   ;SK=10 -- If V0 index = 10 then skip next
  9C         1588    KEYB4 -- Jump to display black key set #2
  9E         1584    KEYB3 -- Jump to display black key set #1

       ; KEYBOARD
       ; BIT PATTERNS

05A0   KEYB5: 8000   ;1 BIT -- Single bit for drawing white keys
  A2   KEYB6: FEFE   ;BLKYS -- Start of black keys set #1
  A4         FE28
  A6         2828   ;  1
  A8         2828
  AA         FEFE   ;BLKYS -- Start of black keys set #2
  AC         FE54
  AE         5454   ;  2
05B0         5454

       ;END KEYBD SUB

       ; **********
       ; * DOORS *
       ; **********

       ; V0, V1 = VX1 VY1
       ; V2, V3 = VX2 VY2

       ; INPUT: NONE
       ; CALLS: TIMER
```

```
05B2  DOORS: 6020   ;V0=20 -- Set up VX, VY for left portion
  B4         6130   ;V1=30 --   of doors
  B6         621F   ;V2=1F -- Set up VX, VY for right portion
  B8         6330   ;V3=30 --   of doors
  BA         A5F2    DOORS -- Set "I" to bit pattern (single bit)
  BC  DOOR1: 70FF   ;V0-1  -- Begin to draw two lines simultaneously
  BE         7201   ;V2+1  --   away from each other

05C0         25F8    DOOR7 -- Do sub to plot two points and delay
  C2         3230   ;SK=30 -- If VX2=30, then top of door done
  C4         15BC    DOOR1 -- Else loop back to complete door top
  C6  DOOR2: 7101   ;V1+1  -- Begin drawing outside vertical lines
  C8         7301   ;V3+1  --   by incrementing both VY coordinates
  CA         25F8    DOOR7 -- Do sub to plot points and delay
  CC         335E   ;SK=5E -- Skip when VY2 at bottommost point
  CE         15C6    DOOR2 -- Else loop back to complete outside lines

05D0  DOOR3: 7001   ;V0+1  -- Begin drawing bottom of doors by
  D2         72FF   ;V2-1  --   adjusting both VX coordinates
  D4         25F8    DOOR7 -- Do sub to plot points and delay
  D6         3220   ;SK=20 -- Skip when VX2 is at center of door
  D8         15D0    DOOR3 -- Else loop back to complete bottom line
  DA  DOOR4: 71FF   ;V1-1  -- Begin drawing center door line by
  DC         73FF   ;V3-1  --   decrementing VY coordinates
  DE         25F8    DOOR7 -- Do sub to plot points and delay

05E0         3331   ;SK=31 -- Skip when center complete
  E2         15DA    DOOR4 -- Else loop to finish center line

     ; DISPLAY HANDLES

  E4         601B   ;V0=1B -- Set up initial VX, VY for displaying
  E6         6143   ;V1=43 --   door handles
  E8         A5F4    DOOR6 -- Set "I" to bit pattern for handle
  EA         D013   ;SHOW  -- Display left door handle
  EC         6023   ;V0=23 -- Set new VX coordinate
  EE         D013   ;SHOW  -- Display right door handle
05F0         00EE   ;RET   -- Return from subroutine

  F2  DOOR5: 8000   ;1 BIT -- Bit for drawing doors
  F4  DOOR6: C0C0   ;HANDL -- Bits for the handles
  F6         C000

     ; DOUBLE SHOW
     ;   AND TIMER CALL  -- Piano doors drawn in "stereo"

  F8  DOOR7: D011   ;SHOW  -- Display one point of door
  FA         D231   ;SHOW  -- Display other point of door
  FC         6F02   ;VF=2  -- Set VF to timer value
  FE         2564    TIMER -- Do sub to delay for time in VF
0600         00EE   ;RET   -- Return from subroutine

     ;END DOORS SUB
```

```
        ; *********
        ; * SYMBL *
        ; *********

        ; V0, V1 = VX, VY
        ; V2     = I INDEX

        ; INPUT : NONE
        ; OUTPUT:
        ;     GCLEF AND
        ;      PRICE DRAWN
        ; CALLS : TIMER

0602  SYMBL: 6201   ;V2=1   -- Will cause "I" to be incremented by 1
  04          6005   ;V0=5   -- V0 is VX for the G-clef symbol
  06          A624    GCLEF -- Set "I" to G-clef bit pattern
  08          2612    SYMB1 -- Do sub to display left symbol on piano
  0A          6033   ;V0=33  -- V0 is VX for the price symbol
  0C          A63C    PRICE -- Set "I" to Price bit pattern (5¢)
  0E          2612    SYMB1 -- Do sub to display right symbol on piano
0610          00EE   ;RET    -- Return from subroutine

        ; DISPLAY SUB

  12  SYMB1: 613B   ;V1=3B -- Set/reset VY for both symbols
  14  SYMB2: 6F02   ;VF=2  -- Set VF to timer value
  16          2564    TIMER -- Do sub to delay for time in VF
  18          D011   ;SHOW  -- Display one bit row of a symbol
  1A          F21E   ;I+V2  -- Increment "I" to next bit row pattern
  1C          7101   ;V1+1  -- Increment VY pointer
  1E          3153   ;SK=53 -- Skip to exit when VY at bottom
0620          1614    SYMB2 -- Else loop to continue drawing
  22          00EE   ;RET    -- Return from subroutine

        ;END SYMBL SUB

        ; GCLEF BITS
  24  GCLEF: 08 08 08 0C 08 0A 08 0A -- Bit patterns for the
  2C         08 0C 18 28 0C 1A 48 09 --   G-clef sign
0634         48 09 08 2A 1C 08 08 08 --

        ; PRICE BITS

  3C  PRICE: AA AA 00 00 E0 E4 84 8E -- Bit patterns for the
0644         88 88 E8 E8 28 28 28 28 --   5¢ price sign
  4C         2E 24 E4 E0 00 00 AA AA --

        ;END PRC/CLF BITS
```

```
        ; *********
        ; * TITLE *
        ; *********

        ; VC, VD = VX, VY
        ; INPUT :
        ;   SCREEN CLEAR
        ; OUTPUT:
        ;   TITLE PRINTED
        ; CALLS: TIMER
        ;      : MESSAGER

0654  TITLE: 6C08  ;VC=8  -- Set up VX, VY coordinates in VC VD
  56         6D1A  ;VD=1A --   for displaying title
  58         A672   ASCI1 -- Set "I" to address ASCII encoded title
  5A  TITL1: 6F20  ;VF=20 -- Set VF to timer value
  5C         2564   TIMER -- Do sub to delay for VF (between letters)
  5E         0244  ;MESGR -- Call messager to print a letter

0660         DCD5  ;PRINT -- DXYN used by messager in interpreter
  62         7C04  ;VC+4  -- Increase VX to next letter position
  64         3C38  ;SK=38 -- Skip when VX past last letter position
  66         165A   TITL1 -- Loop to display "COS-LELODEON"
  68         6C04  ;VC=04 -- Set up new VX VY coordinates
  6A         6D70  ;VD=70 --   for "C. 1979 T. Swan" notice
  6C         0244  ;MESGR -- Call messager to print notice
  6E         DCD5  ;PRINT -- DXYN used by messager in interpreter
0670         00EE  ;RET   -- Return from subroutine

        ;END TITLE

  72  ASCI1: 43 00 4F 00 53 00 2D 00 -- ASCII encoded
  7A         4C 00 45 00 4C 00 4F 00 --   title information
  82         44 00 45 00 4F 00 4E 00 --   "         "
  8A         43 2E 20 31 39 37 39 20 --   "         "
  92         54 2E 53 57 41 4E 00 00 --   "         "

        ; *********
        ; * FIXIT *
        ; *********

        ; VC, VD = VX, VY
        ; V0     = UTILITY
        ; INPUT : TITLE
        ; OUTPUT:
        ;   TITLE FIXED
        ; CALLS: TIMER
        ;      : SHOOT

069A  FIXIT: 6C38  ;VC=38 -- Set up VX, VY coordinates in VC, VD
  9C         6D38  ;VD=38 --   for displaying MICROMAN
  9E         A750   MICRO -- Set "I" to bit pattern for MICROMAN
```

```
06A0  FIXI1: DCDF  ;SHOW  -- Display MICROMAN
  A2         6F08  ;VF=8  -- Set VF to timer value
  A4         2564   TIMER -- Do sub to slow down "walk" into display
  A6         DCDF  ;ERASE -- Erase MICROMAN to animate
  A8         7CFF  ;VC-1  -- Reduce VX coordinate by 1
  AA         3C1A  ;SK=1A -- Skip when VX at desired position
  AC         16A0   FIXI1 -- Loop to walk MICROMAN in
  AE         DCDF  ;SHOW  -- Display stationary figure

06B0         6FA0  ;VF=A0 -- Set VF to timer value
  B2         2564   TIMER -- Do sub to delay before fire sequence
  B4         A760   ARM   -- Set "I" to bits for MICRO's arm
  B6         7D07  ;VD+7  -- Set VY coordinate
  B8         DCD3  ;SHOW  -- Display up arm
  BA         6F40  ;VF=40 -- Set VF to timer value
  BC         2564   TIMER -- Do sub to delay before firing
  BE         60FA  ;V0=FA -- V0 indicates direction to next sub

06C0         270C   SHOOT -- Do sub.  Shoot bad letter
  C2         A760   ARM   -- Set "I" to bits for MICRO's arm
  C4         DCD3  ;ERASE -- Erase arm

     ; FIX TITLE

  C6         6C18  ;VC=18 -- Set up VX, VY in VC VD for
  C8         6D1A  ;VD=1A --   fixing the bad letter
  CA         A764   ASCI2 -- Set "I" to ASCII codes
  CC         0244  ;MESGR -- Call messager
  CE         DCD5  ;PRINT -- DXYN erases bad letter
06D0         0244  ;MESGR -- Call messager
  D2         DCD5  ;PRINT -- DXYN prints right letter

     ; SHOOT C. NOTICE

06D4         6F80  ;VF=80 -- Set up VF to timer value
  D6         2564   TIMER -- Do sub to delay between shootings
  D8         6C1A  ;VC=1A -- Set VX, VY in VC, VD to display
  DA         6D40  ;VD=40 --   MICRO's down arm
  DC         A760   ARM   -- Set "I" to bits for MICRO's arm
  DE         DCD3  ;SHOW  -- Display downward arm

06E0         6F40  ;VF=40 -- Set VF to timer value
  E2         2564   TIMER -- Do sub to delay before firing
  E4         6006  ;V0=06 -- V0 indicates direction to next sub
  E6         270C   SHOOT -- Do sub to shoot downwards
  E8         A760   ARM   -- Set "I" to MICRO's arm bits
  EA         DCD3  ;ERASE -- Erase arm

     ; ERASE BOT 1 PGS    -- See notes on this routine

  EC         6001  ;V0=1  -- Set V0 to # erase pages
  EE         0216  ;PRTCT -- Do MLS to adjust erase
06F0         0200  ;ERASE -- Clear bottom 1 page display
  F2         6004  ;V0=4  -- Set V0 to # erase pages (normal)
  F4         0216  ;PRTCT -- Do MLS to reset erase command
```

```
      ; MOVE MICROMAN
      ; OFF SCREEN

  F6          6D38  ;VD=38 -- Set VY in VD of MICROMAN
  F8          A750   MICRO -- Reset "I" to MICRO's bit patterns
  FA          1702   FIXI3 -- Jump into routine
  FC  FIXI2:  DCDF  ;SHOW  -- Show MICROMAN to animate
  FE          6F08  ;VF=8  -- Set VF to timer value

0700          2564   TIMER -- Do sub to slow down MICRO's exit
  02  FIXI3:  DCDF  ;ERASE -- Erase MICROMAN to animate
  04          7C01  ;VC+1  -- Increase VX coordinate
  06          3C38  ;SK=38 -- Skip when VX at screen edge (right)
  08          16FC   FIXI2 -- Jump to continue exit
  0A          00EE  ;RET   -- Return from sub routine

      ;END FIXIT SUB

      ; *********
      ; * SHOOT *
      ; *********

      ; V0 = DIRECTION
      ; V1, V2 = LOOPS
      ; VC, VD = VX, VY

      ; INPUT :
      ;   V0, VC, VD,
      ; OUTPUT:
      ;   LASER FIRE
      ; CALLS : WOOPS
      ;       : TOGGL

  0C  SHOOT:  A730   SH003 -- Set "I" to laser bit pattern
  0E          0744   TOGGL -- Call MLS to engage hi-speed
0710          6104  ;V1=04 -- Set loop -- number (even!) of shots
  12  SH001:  6208  ;V2=08 -- Set count -- length of fire
  14  SH002:  8D04  ;VD+V0 -- Add direction in V0 to VY
  16          DCD5  ;SHOW  -- Display/erase segment of laser
  18          72FF  ;V2-1  -- Decrease count in V2
  1A          3200  ;SK=0   -- When count is zero, skip next
  1C          1714   SH002 -- Loop to display whole laser
  1E          2736   WOOPS -- Do sub to sound "woop!"

0720          6FFF  ;VF=FF -- These three instructions find
  22          80F3  ;2's -  --   two's compliment of V0 i.e. its
  24          7001  ; comp  --   value subtracted from zero
  26          71FF  ;V1-1  -- Decrease shoot counter V1
  28          3100  ;SK=00 -- If V1=0, then skip to exit
  2A          1712   SH001 -- Else loop to continue firing
  2C          0744   TOGGL -- Call MLS to disengage hi-speed
  2E          00EE  ;RET   -- Return from subroutine
```

```
0730  SH003: 8000  ;GUN    -- Bit pattern for laser fire
  32         8000          --          "           "
  34         8000  ;FIRE   --          "           "

       ;END SHOOT SUB

       ; **********
       ; * WOOPS *
       ; **********

       ; VE = UTILITY

0736  WOOPS: 6E3D  ;VE=3D -- Initialize tone value in VE
  38  WOOP1: BEA7  ;OUT   -- Output VE value to latch
  3A         FE18  ;TONE  -- Set Q for length of VE
  3C         7EFF  ;VE-1  -- Decrease tone value in VE
  3E         3E00  ;SK=0  -- When tone value is zero, skip
0740         1738   WOOP1 -- Else loop to complete woop
  42         00EE  ;RET   -- Return from subroutine

       ;END WOOPS SUB

       ; **********
       ; * TOGGL *
       ; **********

       ; MLS RELOC.
       ; SWITCHES FROM
       ;   HI/LO SPEED

  44  TOGGL: F800          -- Engages/disengages interpreter
  46         BFF8          --   to display at high or
  48         ACAF          --   normal speed.  See issues
  4A         0FFB          --   of VIPER* for details.
  4C         EC5F          -- This sub is completely relocatable
  4E         D400          -- (The final 00 byte is not a part
                           --   of the sub)

       ;END TOGGL MLS

0750  MICRO: 3E 3E 2A 2A 3E 14 1C 00 -- MICROMAN's bit patterns
  58         3E 3E 1C 14 00 36 36 00 --     "                "
0760  ARM:   40 40 40 00             --     "                "
  64  ASCI2: 4C 00 4D 00             -- Bad to good letter codes
```

```
        ; ********
        ; * OPEN *
        ; ********

        ; V0 = LOOP

        ; INPUT :
        ;   DOORS SHUT
        ; OUTPUT:
        ;   DOORS OPEN
        ; CALLS : TIMER
        ;       : RT SLIDE
        ;       : LT SLIDE

0768  OPEN:  600A   ;V0=0A -- Set up loop count in V0
  6A  OPEN1: 6F10   ;VF=10 -- Set VF to timer value
  6C         2564    TIMER -- Do sub to control door speed
  6E         AD8B   ;LEFT  -- Set "I" to display bytes of left door

0770         0918   ;LTSL  -- Call MLS to slide to left (open)
  72         AD8C   ;RIGHT -- Set "I" to display bytes of right door
  74         0900   ;RTSL  -- Call MLS to slide to right (open)
  76         70FF   ;V0-1  -- Decrement loop count in V0
  78         3000   ;SK=0  -- When count is zero, skip to exit
  7A         176A    OPEN1 -- Else loop till doors are open
  7C         00EE   ;RET   -- Return from subroutine

      ;END OPEN SUB

        ; ********
        ; * SHUT *
        ; ********

        ; (SEE OPEN)
        ; INPUT :
        ;   DOORS OPEN
        ; OUTPUT:
        ;   DOORS SHUT

  7E  SHUT:  600A   ;V0=0A -- Set up loop count in V0
0780  SHUT1: 6F10   ;VF=10 -- Set timer value in VF
  82         2564    TIMER -- Do sub to control door speed
  84         AD8A   ;LEFT  -- Set "I" to display bytes of left door
  86         0900   ;RTSL  -- Call MLS to slide to right (close)
  88         AD8D   ;RIGHT -- Set "I" to display bytes of right door
  8A         0918   ;LTSL  -- Call MLS to slide to left (close)
  8C         70FF   ;V0-1  -- Decrement loop count in V0.
  8E         3000   ;SK=0  -- When count is zero, skip to exit
0790         1780    SHUT1 -- Else loop to close doors
  92         00EE   ;RET   -- Return from subroutine

      ;END SHUT SUB
```

```
        ; ********
        ; * PLAY *
        ; ********

        ; V0 = PITCH
        ; V1 = DURATION
        ; V2 = KEY NUMBER

        ;CALLS: INIT
        ;      : NEXT
        ;      : SCROL
        ;      : PUNCH

0794  PLAY:  AA00  ;SONG  -- Set "I" to note list minus 3 bytes
  96         096C  ;INIT  -- Call MLS to initialize note pointer
  98  PLAY1: 0977  ;NEXT  -- Call MLS to set "I" to next note
  9A         F265  ;GET   -- Set up V0, V1, V2 from note list
  9C         0930  ;SCROL -- Call MLS to scroll paper
  9E         40FF  ;SK≠FF -- If pitch ≠ FF then skip

07A0         00EE  ;RET   -- Return on pitch = FF = end of song
  A2         8300  ;V3-V0 -- Save pitch in V3 from V0
  A4         4000  ;SK≠00 -- If pitch ≠ 0 (not rest) skip next
  A6         17CC   PLAY2 -- Jump to skip playing a rest note

        ; FIGURE VX, VY

07A8         A989  ;KEYTB -- Set "I" to key table address
  AA         F21E  ;I+V2  -- Add note code to index table
  AC         F065  ;GET   -- Let V0 = byte from table (VX coord.)
  AE         6F1E  ;VF=1E -- VF holds constant of hex 1E

07B0         8F25  ;VF-V2 -- Subtract note code.  If neg., then V2>1E
  B2         A7E2   BLACK -- Set "I" to black key bits
  B4         6276  ;V2=76 -- Set V2 to VY coordinate black keys
  B6         3F00  ;SK=00 -- Skip if the subtraction was negative
  B8         A7DC   WHITE -- Change "I" to white key bits
  BA         3F00  ;SK=00 -- Continue to skip only if negative
  BC         627C  ;V2=7C -- Change V2 to VY for white keys

        ; STRIKE NOTE
        ;   V3 HOLDS PITCH

  BE         0930  ;SCROL -- Call MLS to continue scrolling
07C0         CF03  ;RNDOM -- Let VF = random byte from 0-3
  C2         3F00  ;SK=00 -- Skip next 25% of time
  C4         095E  ;PUNCH -- Call MLS to punch paper with holes
  C6         D025  ;KEY   -- Depress appropriate key by erasing
  C8         B3A7  ;OUT   -- Output pitch in V3 to latch
  CA         F118  ;TONE  -- Set Q to switch on Simple Sound
  CC  PLAY2: F115  ;TIMER -- Set timer to same value
  CE         0930  ;SCROL -- Call MLS to continue scrolling
```

```
07D0  PLAY3: F107   ;V1=T   -- Let V1 = current timer value
  D2         3101   ;SK=1   -- When timer not quite zero, skip
  D4         17D0    PLAY3 -- Jump to continue playing
  D6         3300   ;SK=0   -- If pitch was zero, skip next
  D8         D025   ;KEY    -- Show to release key depressed
  DA         1798    PLAY1 -- Jump to play next note

       ; BIT PATTERNS

  DC  WHITE: C0 C0 C0 C0 00 00 -- White key bit pattern
07E2  BLACK: 80 80 80 80 80 00 -- Black key bit pattern

       ;END PLAY SUB

07E8 - 07FF -- NOT USED -- May be set to all zeros

       ; **********
       ; * PIANO *
       ; **********

       ;V0, V1 = VX, VY
       ;V2, V3 = DIRECTION

       ; INPUT : NONE
       ; OUTPUT:
       ;   DRAWS BODY OF
       ;   PIANO

0800  PIAN1: 8000   ;1 BIT -- Bit for drawing piano body

  02  PIANO: A800    PIAN1 -- Set "I" to bit pattern above
  04         6000   ;V0=0   -- Set up initial VX, VY in V0, V1
  06         616F   ;V1=6F  --   to begin drawing piano

       ; LEFT CORNER

  08         6201   ;V2=1   -- Set up initial directions in
  0A         63FE   ;V3=-2  --   variables V2, V3 (45 degrees)
  0C  PIAN2: 2868    PIANA -- Do sub to plot point and delay
  0E         3003   ;SK=3   -- Skip to next part when V0 = 3 (VX)
0810         180C    PIAN2 -- Loop to complete left corner

       ; LEFT SIDE

  12         6200   ;V2=0   -- Set new directions in
  14         63FF   ;V3=-1  --   variables V2, V3 (0 degrees)
  16  PIAN3: 2868    PIANA -- Do sub to plot point and delay
  18         3110   ;SK=10  -- Skip to next part when V1=10 (VY)
  1A         1816    PIAN3 -- Loop to complete left side

       ; TOP (FRONT)
```

```
081C          6201  ;V2=1   -- Set new directions in
  1E          6300  ;V3=0   --   variables V2, V3 (90 degrees)
0820  PIAN4:  2868   PIANA  -- Do sub to plot point and delay
  22          303B  ;SK=3B  -- Skip to next part when V0 = 3B (VX)
  24          1820   PIAN4  -- Loop to complete top (front)

      ; RIGHT SIDE

0826          6200  ;V2=0   -- Set up new directions in
  28          6301  ;V3=1   --   variables V2, V3 (180 degrees)
  2A  PIAN5:  2868   PIANA  -- Do sub to plot point and delay
  2C          3169  ;SK=69  -- Skip to next part when V1 = 69 (VY)
  2E          182A   PIAN5  -- Loop to complete right side

      ; RIGHT CORNER

0830          6201  ;V2=1   -- Set up new directions in
  32          6302  ;V3=2   --   variables V2, V3 (135 degrees)
  34  PIAN6:  2868   PIANA  -- Do sub to plot point and delay
  36          303F  ;SK=3F  -- Skip to next part when V0 = 3F (VX)
  38          1834   PIAN6  -- Loop to complete right corner

      ; DEPTH LINE

  3A          6004  ;V0=4   -- Set up new VX VY coordinates
  3C          6169  ;V1=69  --   in variables V0, V1
  3E          6201  ;V2=1   -- Set up new direction
0840          6300  ;V3=0   --   in variables V2, V3 (90 degrees)
  42  PIAN7:  2868   PIANA  -- Do sub to plot point and delay
  44          303B  ;SK=3B  -- Skip to next part when V0 = 3B (VX)
  46          1842   PIAN7  -- Loop to complete line above keyboard

      ; TOP (REAR)

  48          6003  ;V0=3   -- Set up new VX, VY coordinates
  4A          610F  ;V1=0F  --   in variables V0, V1
  4C          6201  ;V2=1   -- Set up new direction
  4E          63FE  ;V3=-2  --   in variables V2, V3 (45 degrees)
0850  PIAN8:  2868   PIANA  -- Do sub to plot point and delay
  52          3006  ;SK=6   -- Skip to next part when V0 = 06 (VX)
  54          1850   PIAN8  -- Loop to continue top left conrer
  56          6300  ;V3=0   -- Set up new direction (180 degrees)
  58  PIAN9:  2868   PIANA  -- Do sub to plot point and delay
  5A          3038  ;SK=38  -- Skip to next part when V0 = 38 (VX)
  5C          1858   PIAN9  -- Loop to complete top (back)

  5E          6302  ;V3=2   -- Set up new direction (135 degrees)
0860  PIA10:  2868   PIANA  -- Do sub to plot point and delay
  62          303C  ;SK=3C  -- Skip to exit when V0 = 3C (VX)
  64          1860   PIA10  -- Loop to complete top right corner
  66          00EE  ;RET    -- Return from subroutine

      ;END PIANO SUB
```

```
        ; *********
        ; * PIANA *
        ; *********

        ; INPUT :
        ;   V0, V1, V2, V3
        ;   I = POINT
        ; OUTPUT:
        ;   POINT PLOTTED
        ;   V0 = V0 + V2
        ;   V1 = V1 + V3
        ; CALLS : TIMER

0868  PIANA: 6F02  ;VF=2  -- Set VF to timer value
  6A          2564   TIMER -- Do sub to delay between points
  6C          D011  ;SHOW  -- Display point at V0, V1
  6E          8024  ;V0+V2 -- Add V2 direction vector to V0 (VX)
0870          8134  ;V1+V3 -- Add V3 direction vector to V1 (VY)
  72          00EE  :RET   -- Return from subroutine

        ;END PIANA SUB

        ; *********
        ; * ENCOR *
        ; *********

        ; V0 = VX, UTILITY
        ; V1 = VY

        ; INPUT : DOORS
        ;   SHUT
        ; OUTPUT: MICRO-
        ;   MAN TAKES
        ;   A BOW

0874  ENCOR: 6FA0  ;VF=A0 -- Set VF to timer value
  76          2564   TIMER -- Do sub to delay encore

        ; OPEN DOORS

  78          6005  ;V0=5  -- Set V0 to open doors half way
  7A          276A   OPEN1 -- Do sub opening doors

        ; SHOW MICROMAN

  7C          601B  ;V0=1B -- Set VX, VY coordinates in
  7E          6140  ;V1=40 --   variables V0, V1
0880          A750   MICRO -- Set "I" to bits for MICROMAN
  82          D01F  ;SHOW  -- Display MICROMAN at V0, V1
  84          6F40  ;VF=40 -- Set Timer value in VF
  86          2564   TIMER -- Do sub to delay winking
```

```
        ; WINK RIGHT EYE

0888            A8A4    LTEYE -- Set "I" to bits for MICRO's eye
  8A            D014    ;SHOW -- Display to wink eye
  8C            2736    WOOPS -- Do sub sounding a "woop!"
  8E            6F10    ;VF=10 -- Set VF to timer value
0890            2564    TIMER -- Do sub to delay opening eye
  92            D014    ;ERASE -- Erase to re-open eye
  94            2736    WOOPS -- Do sub sounding second "woop!"

        ; ERASE MICROMAN

0896            6F40    ;VF=40 -- Set VF to timer value
  98            2564    TIMER -- Do sub to delay before end
  9A            A750    MICRO -- Set "I" to MICROMAN's bits
  9C            D01F    ;ERASE -- Erase MICROMAN

        ; CLOSE DOORS

  9E            6005    ;V0=5  -- Set V0 to close half open doors
08A0            2780    ;SHUT1 -- Do sub to close the doors
  A2            00EE    ;RET   -- Return from subroutine

  A4    LTEYE:  0000            -- Bit patterns for MICROMAN's
  A6            0404              left eye

        ;END ENCOR

08A8 - 08FF -- NOT USED -- May be set to all zeros
```

RIGHT SLIDE (either door)

```
0900  00        RTSLD: IDL          ;Wait for interrupt to occur
  01  F8 2D             LDI  $2D    ;Loop count = door height
  03  AF                PLO  RF     ;RF.0 is loop counter
  04  0A           1H:  LDN  RA     ;Load byte from display
  05  F6                SHR         ;Shift right one bit
  06  5A                STR  RA     ;Return byte to display
  07  1A                INC  RA     ;Increment pointer (RA)
  08  0A                LDN  RA     ;Load second byte from display
  09  76                SHRC        ;Shift right with carry
  0A  5A                STR  RA     ;Return byte to display
  0B  8A                GLO  RA     ;Add 7 to RA to point
  0C  FC 07             ADI  $07    ;  down one row
  0E  AA                PLO  RA     ;
  0F  9A                GHI  RA     ;Add possible carry to

0910  7C 00             ADCI $00    ;  RA.1
  12  BA                PHI  RA     ;
  13  2F                DEC  RF     ;Decrement loop counter RF
  14  8F                GLO  RF     ;Test loop count
  15  3A 04             BNZ  1B     ;If ≠, branch to loop
  17  D4                SEP  R4     ;Return.  Door shifted
                                    ;  one bit to the right
```

LEFT SLIDE

```
0918  00        LTSLD: IDL          ;Wait for interrupt to occur
  19  F8 2D             LDI  $2D    ;This sub works identically
  1B  AF                PLO  RF     ;  to RTSLD except that
  1C  0A           1H:  LDN  RA     ;  the door will be shifted
  1D  FE                SHL         ;  one bit to the left
  1E  5A                STR  RA     ;
  1F  2A                DEC  RA     ;Both subs use RA to

0920  0A                LDN  RA     ;  address display bits.  RA
  21  7E                SHLC        ;  is set by the CHIP-8
  22  5A                STR  RA     ;  instruction, AMMM.
  23  8A                GLO  RA
  24  FC 09             ADI  $09    ;Note that these two
  26  AA                PLO  RA     ;  subs simply shift a
  27  9A                GHI  RA     ;  block of memory two
  28  7C 00             ADCI $00    ;  bytes wide and 2D bytes
  2A  BA                PHI  RA     ;  deep (one door) allowing
  2B  2F                DEC  RF     ;  the piano doors to appear
  2C  8F                GLO  RF     ;  to open then close.
  2D  3A 1C             BNZ  1B
  2F  D4                SEP  R4
```

SCROLL SUB

```
0930  F8 2A     SCROL: LDI  $2A    ;Load loop count of $2A into R7.0
  32  A7               PLO  R7     ;
  33  F8 0D            LDI  $0D    ;Set up display pointers RE & RF
  35  BE               PHI  RE     ;  with RF just "below" RE
  36  BF               PHI  RF     ;  at top edge of paper roll
  37  F8 93            LDI  $93    ;
  39  AE               PLO  RE     ;RE = $0D93
  3A  F8 9B            LDI  $9B    ;
  3C  AF               PLO  RF     ;RF = $0D9B
  3D  4F          1H:  LDA  RF     ;Get a byte from below (RF+1)
  3E  5E               STR  RE     ;Store above
  3F  1E               INC  RE     ;Increment RE (RF was incremented)

0940  0F               LDN  RF     ;Get second byte from below
  41  5E               STR  RE     ;Store above
  42  8E               GLO  RE     ;Add 7 to RE using double
  43  FC 07            ADI  $07    ;  precision to address
  45  AE               PLO  RE     ;  next row of bits in
  46  9E               GHI  RE     ;  the paper roll
  47  7C 00            ADCI $00    ;
  49  BE               PHI  RE     ;
  4A  8F               GLO  RF     ;Add 7 to RF using double
  4B  FC 07            ADI  $07    ;  precision to address
  4D  AF               PLO  RF     ;  next row of bits in
  4E  9F               GHI  RF     ;  the paper roll
  4F  7C 00            ADCI $00    ;

0951  BF               PHI  RF     ;
  52  27               DEC  R7     ;Decrement loop count in R7.0
  53  87               GLO  R7     ;Test loop count
  54  3A 3D            BNZ  1B     ;If ≠ 0, then branch to continue
  56  D4               SEP  R4     ;Return from subroutine
```

0957 - 095D -- NOT USED -- May be set to all zeros

PUNCH SUB

```
  5E  F8 0E     PUNCH: LDI  $0E    ;Set RF to the address
0960  BF               PHI  RF     ;  of last row of piano
  61  F8 DC            LDI  DC     ;  "paper roll"
  63  AF               PLO  RF     ;RF = 0EDC
  64  EF               SEX  RF     ;X = F
  65  99               GHI  R9     ;Get R9.1 (incremented in interrupt)
  66  F3               XOR         ;Combine with display via XOR
  67  73               STXD        ;Store in display (RF - 1)
  68  89               GLO  R9     ;Get R9.0 (incremented in interrupt)
  69  F3               XOR         ;Combine with display
  6A  5F               STR  RF     ;Store in display
  6B  D4               SEP  R4     ;Return -- paper "punched"
```

INITIALIZE NOTE LIST

```
096C  93          INIT: GHI  R3     ;Set RF to address of two
 6D   BF                PHI  RF     ;  byte variable located
 6E   F8 87             LDI  $87    ;  in NEXT sub at 0987
0970  AF                PLO  RF
 71   9A                GHI  RA     ;Store RA at this address
 72   5F                STR  RF     ;  RA was set by a CHIP-8
 73   1F                INC  RF     ;  instruction to the location
 74   8A                GLO  RA     ;  of the note table.  This
 75   5F                STR  RF     ;  sub remembers that location
 76   D4                SEP  R4     ;Return
```

NEXT NOTE SUB

```
0977  93          NEXT: GHI  R3     ;Set RF to the NXNOTE
 78   BF                PHI  RF     ;  variable below at
 79   F8 88             LDI  $88    ;  $0988.
 7B   AF                PLO  RF     ;
 7C   0F                LDN  RF     ;Get low byte of variable
 7D   FC 03             ADI  $03    ;Add 3 to address next note
 7F   AA                PLO  RA     ;Put in RA.0 (CHIP-8 "I" pointer)

0980  2F                DEC  RF     ;Point to high byte NXNOTE
 81   0F                LDN  RF     ;Get high byte of variable
 82   7C 00             ADCI $00    ;Add possible carry
 84   BA                PHI  RA     ;Put in RA.1 (CHIP-8 "I" pointer)
 85   30 72             BR   $72    ;Branch to reset variable
 87   00          NXNOTE: BYTE $00  ;These two bytes always
 88   00                  BYTE $00  ;  contain the address
                                    ;  of the next note to
                                    ;  be played
```

KEY TABLE

```
0989  00 02 04 06 08 0A 0C          ;This table contains the X
0990  0E 10 12 14 16 18 1A 1C       ;  coordinates of each key
 98   1E 20 22 24 26 28 2A 2C       ;  for display.  The note code
 A0   2E 30 32 34 36 38 3A 3C       ;  plus the address 0989 will
 A8   02 04 08 0A 0C 10 12 16       ;  get the proper X coordinate
 B0   18 1A 1E 20 24 26 28 2C       ;  for white or black keys
 B8   2E 32 34 36 3A 3C             ;  (Black starts @ 09A8)
```

09BE - 09FF -- NOT USED -- May be set to all zeros

ALLEY CAT FOR COS-MELODEON

```
0A03  2B 06 2A  D#
  06  29 06 10  E
  09  22 06 12  G
  0C  1E 06 13  A
  0F  19 06 15  C

0A12  00 12 00  (r)
  15  1B 08 14  B
  18  00 12 00  (r)
  1B  1E 08 13  A
  1E  00 12 00  (r)

0A21  20 08 2C  G#
  24  00 12 00  (r)
  27  22 08 12  G
  2A  00 12 00  (r)
  2D  20 08 2C  G#

0A30  00 06 00  (r)
  33  1E 28 13  A
  36  00 0F 00  (r)
  39  22 0D 12  G
  3C  00 0B 00  (r)
  3F  22 08 12  G

0A42  20 08 2C  G#
  45  00 12 00  (r)
  48  1E 08 13  A
  4B  00 12 00  (r)
  4E  1D 08 2D  A#

0A51  00 12 00  (r)
  54  1B 28 14  B
  57  00 04 00  (r)
  5A  49 08 26  F#
  5D  45 0F 0B  G

0A60  8B 20 04  G
  63  00 04 00  (r)
  66  1B 08 14  B
  69  00 12 00  (r)
  6C  19 08 15  C
  6F  00 12 00  (r)

0A72  1B 08 14  B
  75  00 12 00  (r)
  78  1E 08 13  A
  7B  00 12 00  (r)
  7E  22 08 12  G

0A81  00 12 00  (r)
  84  20 08 2C  G#
  87  00 06 00  (r)
  8A  1E 28 13  A
  8D  00 0F 00  (r)

0A90  22 0D 12  G
  93  00 0B 00  (r)
  96  22 08 12  G
  99  20 08 2C  G#
  9C  00 12 00  (r)
  9F  1E 08 13  A

0AA2  00 12 00  (r)
  A5  1B 08 14  B
  A8  00 12 00  (r)
  AB  19 10 15  C
  AE  94 08 21  F#

0AB1  8B 0F 04  G
  B4  D2 18 00  C
  B7  00 0F 00  (r)
  BA  19 08 15  C
  BD  00 12 00  (r)

0AC0  16 08 16  D
  C3  00 12 00  (r)
  C6  16 03 16  D
  C9  2E 03 0F  D
  CC  16 03 16  D
  CF  2E 03 0F  D

0AD2  16 03 16  D
  D5  2E 03 0F  D
  D8  16 03 16  D
  DB  2E 03 0F  D
  DE  16 03 16  D
```

5.01.26

```
0AE1  00 08 00  (r)
  E4  19 0D 15  C
  E7  00 04 00  (r)
  EA  16 03 16  D
  ED  2E 03 0F  D

0AF0  16 03 16  D
  F3  2E 03 0F  D
  F6  16 03 16  D
  F9  2E 03 0F  D
  FC  16 03 16  D
  FF  2E 03 0F  D

0B02  16 03 16  D
  05  2E 03 0F  D
  08  16 03 16  D
  0B  00 18 00  (r)
```

0B0E-0B5E -- Repeat from 0ABA

```
0B5F  00 1A 00  (r)
  62  1B 06 14  B
  65  19 06 15  C
  68  18 06 2E  C#
  6B  16 1A 16  D
  6E  19 1A 15  C

0B71  1B 1A 14  B
  74  1E 1A 13  A
  77  22 1A 12  G
  7A  24 10 2B  F#
  7D  26 1D 11  F
```

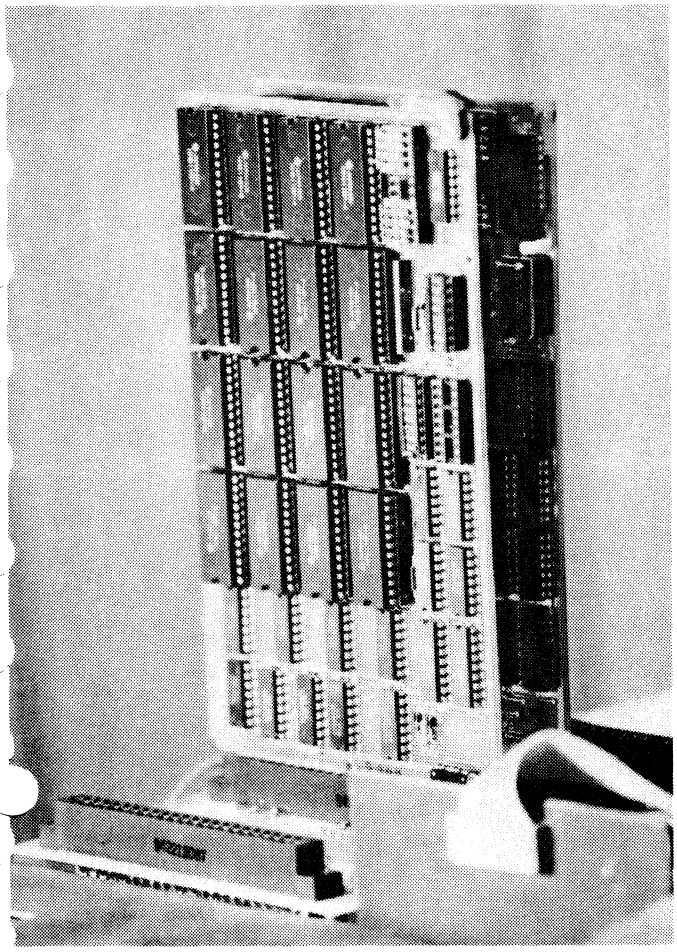
0B80-0BB2 -- Repeat from 0A03 to 0A36

```
0BB3  00 0F 00  (r)
  B6  24 08 2B  F#
  B9  22 38 12  G
  BC  20 38 2C  G#
  BF  1E 38 13  A

0BC2  1B 38 14  B
  C5  19 06 15  C
  C8  00 08 00  (r)
  CB  94 08 21  F#
  CE  8B 0F 04  G

0BD1  D2 06 00  C
  D4  00 18 00  (r)
  D7  19 06 15  C
  DA  FF        (end)
```

NOW - - - 64K MEMORY AVAILABLE FOR YOUR VIP ! ! ! !



Introductory Special . . .

VSP626 32K ROM/RAM card with 4K of 6116 RAM installed plus one VSP001 expander/adapter: $149 + shipping

Same fully stuffed with 32K of 6116 RAM: $249 + shipping

Your VSP626 will be delivered set up for the lower half of memory space, i.e., ØØØØ-7FFFH. Memory addresses 7ØØØ-7FFF can be inhibited to permit use of on-board VIP RAM with your VSP626.

Prices subject to change without notice.
Pennsylvania residents add 6% sales tax.

VIP Consultants:

George S. Gadbois, P.E.
David E. Van Zandt
P.O. Box 7062
Lancaster, PA 17604

Two VSP626 boards and VSP001 expander/adapter installed in VIP for 64K memory capability.

The VSP626 32K ROM/RAM card is a versatile memory module having on-board address latches and decoders. Address latches and data lines are buffered to minimize loading of the VIP system expansion bus. The VSP626 card contains 16 24-pin sockets which can be populated with 2716 EPROMs or 6116 CMOS RAMs or mixed ROM and RAM. A VSP001 expander/adapter is required to use the VSP626 memory card with the VIP.

The VSP001 bus expander/adapter provides one socket for expansion cards using the VIP system bus and four sockets for the VSP626 memory cards and other function cards to be announced soon. No modification of your VIP is required to add one VSP626 card for memory addresses ØØØØ to 7FFFH. Expansion of VIP memory above 8ØØØH requires that the VIP system ROM be removed and a new operating system be installed in your VSP626 memory card.

We are currently working on a new operating system for the VIP using ASCII keyboard input. We will be offering the new operating system installed in a VSP626 card for full 64K capability in your VIP in the near future.